

Frame-Logik (F-Logic)

Pascal Hitzler
York Sure, Anupriya Ankolekar
Folien von Markus Krötzsch

Institut für Angewandte Informatik und
Formale Beschreibungsverfahren (AIFB)

Folien zur Vorlesung
Intelligente Systeme im WWW

SS 2006

- Beschreibungssprachen wie OWL oft „**eigenschaftszentriert**“:
Relationen (Rollen) und Klassen als Grundlage zur
Klassifizierung von Instanzen
- Programmierung typischerweise **objektorientiert**:
Relationen (Eigenschaften) sind Klassen zugeordnet,
Instanzen als komplex strukturierte Objekte

Zielstellung

Objektorientierte Ontologiesprache F-Logik

Weitere Features

- Closed World Semantics (wie in Datenbanken)
- Logische Regeln (wie in Logikprogrammierung)

Beispiel

```
/* Fakten */
iokaste:frau.
laios:mann.
oedipus:mann[vater->laios; mutter->iokaste].
polyneikes:mann[vater->oedipus; mutter->iokaste;
                schwester->>{antigone, ismene};
                bruder->>eteokles:mann]

/* Regeln */
FORALL X,Y  X[sohn->>Y]  <-  Y:mann[vater->X].
FORALL X,Y  X[sohn->>Y]  <-  Y:mann[mutter->X].

/* Anfragen */
FORALL X,Y  <-  X:mann[sohn->>Y[mutter->iokaste]].
```

Objekte, Methoden und Parameter

- *Objekte* haben eindeutige Bezeichner:
iokaste:frau. laios:mann. ...
↔ **Unique Name Assumption**
- *Objekteigenschaften* ↔ *Methoden*
Einwertige Methoden:
oedipus[vater->laios].
Mehrwertige Methoden:
polyneikes[schwester->>{antigone, ismene}].
- *Parameter* für Methoden:
iokaste[sohn@(laios)->oedipus;
sohn@(oedipus)->{polyneikes, eteokles}].

Klassen und Subklassen

- Objekte können Klassen zugewiesen werden:

```
iokaste:frau.  
laios:mann.
```

- Klassen können hierarchisch geordnet werden:

```
frau::person.  
mann::person.
```

Einbettung (Nesting)

Beschreibungen von Objekten können geschachtelt werden:

```
eteokles:mann[vater->oedipus:mann  
  [vater->laios; mutter->iokaste];  
  mutter->iokaste:frau].
```

Atome und Moleküle

- Atome:

```
iokaste:frau.  laios[vater->labdacus].
```

- Moleküle:

```
eteokles:mann[vater->oedipus;  
  mutter->iokaste;  
  schwester->>{antigone, ismene};  
  bruder->>polyneikes].
```

- Moleküle können in Atome zerlegt werden:

```
eteokles:mann.  
eteokles[vater->oedipus].  
eteokles[mutter->iokaste].  
eteokles[schwester->>antigone].  
eteokles[schwester->>ismene].  
eteokles[bruder->>polyneikes].
```

Signaturen

- Beschreibung der zulässigen Methoden in Klassen

- Beispiel:

```
person[mutter=>frau].  
person[vater=>mann].  
person[tochter=>>frau].  
frau[sohn@(mann)=>>mann].
```

- Typenprüfung möglich

Prädikate

- Beschreibung von Eigenschaften/Beziehungen von Objekten und Klassen:

```
mörder(oedipus, laios).  
selbstmörderin(iokaste).  
mörder(eteokles, polyneikes).  
mörder(polyneikes, eteokles).  
selbstmörderin(antigone).  
...
```

- Beliebige Stelligkeit (Vergleich DL):

```
tochterVon(iokaste, oedipus, antigone).  
aussage.
```

Regeln

- Regeln sind Implikationen:

```
FORALL X selbstmörderin(X) <-  
    frau(X) AND mörder(X, X).  
FORALL X, Y X[schwester->Y] <-  
    EXISTS Z X[mutter->Z] AND Y[mutter->Z].  
FORALL X unhappy_man(X) <-  
    EXISTS Y, V, W X:mann[tochter->Y; sohn->{V, W}]  
    AND (mörder(V, W) OR selbstmörderin(Y)).
```

- Unterteilung in Kopf und Rumpf
- Beliebige logische Ausdrücke im Rumpf

Anfragen

- Anfragen sind Regeln ohne Kopf:

```
FORALL X, Y <- X:mann[vater->oedipus;  
    schwester->Y] AND selbstmörder(Y).
```

- Anfrageergebnis: Variablenbindungen, die Bedingung erfüllen
- **Closed World Assumption** (siehe Semantik)
- Quantifikation über Klassen ist zulässig:

```
FORALL X, C <- X:C[vater->oedipus].
```

Semantik von F-Logik

Was *genau* bedeuten F-Logik-Spezifikationen?

Grundidee

Umwandlung von F-Logik in Logikprogramme

Problem

Auch die Semantik von (ausdrucksstarken) Logikprogrammen ist kompliziert

F-Logik → Logikprogramm (1)

- Umwandlung von Molekülen in Atome (siehe oben)

- Umwandlung von Atomen in Prädikate:

```
a:C      ↦ isa_(a,C)
A::B     ↦ sub_(A,B)

a[B→c]  ↦ att_(a,B,c)
a[B⇒c]  ↦ setatt_(a,B,c)

A[B⇒C]  ↦ atttype_(A,B,C)
A[B⇒C]  ↦ setatttype_(A,B,C)
```

Beispiel

```
FORALL X,Y,Z X[bruder→Y] <-
              X[vater→Z] AND Y[vater→Z].
```

⇓

```
FORALL X,Y,Z setatt_(X,bruder,Y) <-
              att_(X,vater,Z) AND att_(Y,vater,Z).
```

F-Logik → Logikprogramm (2)

- Was bedeutet „sub_(A, B)“ etc.?

Hilfsprädikate müssen durch Axiome beschrieben werden!

- Beispiel: // Deklarierte Eigenschaften werden vererbt:

```
FORALL X,Y,A,B atttype_(X,A,B) <-
                atttype_(Y,A,B) AND sub_(X,Y).
```

// Subklassenbeziehung transitiv:

```
FORALL X,Y,Z sub_(X,Z) <-
                sub_(X,Y) AND sub_(Y,Z).
```

...

F-Logik → Logikprogramm (3)

- Regeln in einem allgemeinen Logikprogramm:

```
H <- B1, NOT B2, NOT B3, B4, B5.
```

- F-Logik erlaubt kompliziertere Ausdrücke im Rumpf:

```
FORALL X größtes(X) <- p(X) AND
                FORALL Y (p(Y)→ kleingleich(Y,X)).
```

- ⇔ **Lloyd-Topor-Transformation:**

⇓

```
FORALL X,Y größtes(X) <- p(X) AND
                (NOT p(Y) OR kleingleich(Y,X)).
```

⇓

```
FORALL X größtes(X) <- p(X) AND NOT q(X).
FORALL X,Y q(X) <- p(Y) AND
                NOT kleingleich(Y,X).
```

Auswertung von Logikprogrammen

- Definite Logikprogramme (ohne Negation):
Kleinstes Modell \Leftrightarrow SLD-Resolution

Problem

Erzeugung von Logikprogrammen aus F-Logik kann zu Negationen führen.

\rightsquigarrow Eventuell kein kleinstes Modell möglich

- Beispiel:

```
person(gerrit).  
mann(X) <- person(X) AND NOT frau(X).
```

Erlaubt zwei minimale Modelle:

{person(gerrit), frau(gerrit)} und
{person(gerrit), mann(gerrit)}

Beispiel

Es gibt Programme, die nicht stratifizierbar sind!

```
person(gerrit).  
mann(X) <- person(X) AND NOT frau(X).  
frau(X) <- person(X) AND NOT mann(X).
```

\rightsquigarrow Kompliziertere Semantik nötig!

Stratifizierung

Idee

Teile Programm in „Schichten“, die nacheinander ausgewertet werden.

- Beispiel:

```
person(gerrit). Stratum 0  
-----  
mann(X) <- person(X) AND NOT frau(X). Stratum 1
```

\rightsquigarrow „Perfect Model“ {person(gerrit), mann(gerrit)}

- Negative Bedingungen im Rumpf dürfen nur in tieferen Schichten definiert sein, positive außerdem auch auf der selben Schicht.

Well-founded Semantics

Idee

Logische Aussagen können *wahr*, *falsch*, aber auch *unbekannt* sein.

\rightsquigarrow „Mache nur das falsch, was keinesfalls wahr ist. Mache nur das wahr, was mit Sicherheit wahr ist.“

- Verschiedene äquivalente formale Definitionen möglich

Beispiel

```
person(gerrit).  
mann(X) <- person(X) AND NOT frau(X).  
frau(X) <- person(X) AND NOT mann(X).
```

WFS-Modell: {person(gerrit)}

```
mann(ulf).  
mann(barbier).  
rasiert(barbier,X) <- mann(X) AND  
                        NOT rasiert(X,X).
```

WFS-Modell: {mann(ulf), mann(barbier),
rasiert(barbier,ulf), ¬rasiert(ulf,ulf),
¬rasiert(ulf,barbier)}

Eigenschaften der Well-founded Semantics

- Für definite Programme liefert WFS das kleinste Modell
- Für stratifizierte Programme liefert WFS das „Perfect Model“
- Anwendbar auf alle *normalen* Logikprogramme
- Hochgradig unentscheidbar.

Vor- und Nachteile von F-Logik

Vorteile

- Große Ausdrucksstärke (Regeln, Signaturen, ...)
- Objektorientiert
- Leistungsstark im Umgang mit Instanzen

Nachteile

- Keine klassische Logik
- Verschiedene mögliche Semantiken
- Hochgradig unentscheidbar

F-Logik vs. DL

Wichtigste Unterschiede zu Beschreibungslogiken:

F-Logik

Objektorientiert

Klassen und Instanzen
als FOL-Terme

Closed world

Unentscheidbar

Unique name assumption

Nicht standardisiert

DL

„Eigenschaftsorientiert“

Instanzen als FOL-Terme
Klassen als FOL-Prädikate

Open world

(Oft) entscheidbar

Keine UNA

offizieller Standard OWL DL

- Weitere Features: Pfadausdrücke für Eigenschaften (wie in OO-Programmierung), Datenbankbindung, Namensräume, etc.
- Implementiert z.B. im **OntoBroker** der *ontoprise* GmbH
- Praxisorientierte Algorithmen für Annäherung der formalen Semantik